**MACHINE LEARNING**

**DATASET:** **Bank Loan Status Dataset**

**NAME: R.KARTHIKEYAN**

**ROLL NO: 195214123**

**CLASS: I M.Sc-CS-A**

**SHIFT: I**

**MACHINE LEARNING PROJECT**

In this project, I have used Linear Regression Machine Learning model for the **Bank Loan Status Dataset** from Kaggle website

**PROBLEM STATEMENT**

Each record in the database describes Future Loan Status prediction via classification models. The Bank Loan Status Dataset was drawn from the Kaggle website. The attributes are defined as include them here along with any citations of past research.

**ML METHODOLOGY**

Linear Regression is the methodology used for training and testing the dataset. Linear Regression is a method of modeling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables. Linear Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.

**DATASET DESCRIPTION**

**Columns:**

Loan ID, Customer ID, Current Loan Amount, Term, Credit Score, Annual Income, Years in current job, Home Ownership, Purpose, Monthly\_Debt, Years of Credit History, Months since last delinquent, Number of Open Accounts, Number of Credit Problems, Current Credit Balance, Maximum Open Credit, Bankruptcies, Tax Liens.

**PRE-PROCESSING**

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.

**# Matplotlib visualization**

import matplotlib.pyplot as plt

%matplotlib inline

**# Set default font size**

plt.rcParams['font.size'] = 24

**# Internal ipython tool for setting figure size**

from IPython.core.pylabtools import figsize

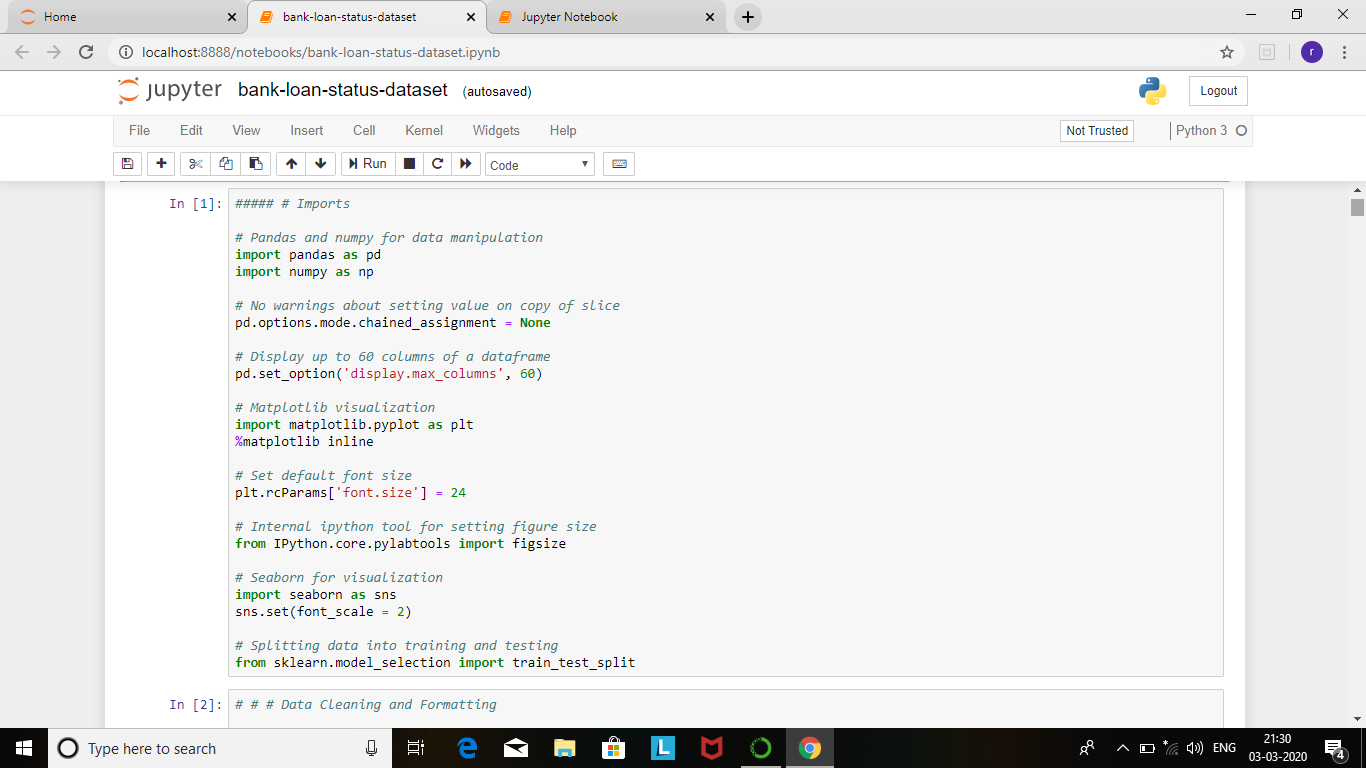
**# Seaborn for visualization**

import seaborn as sns

sns.set(font\_scale = 2)

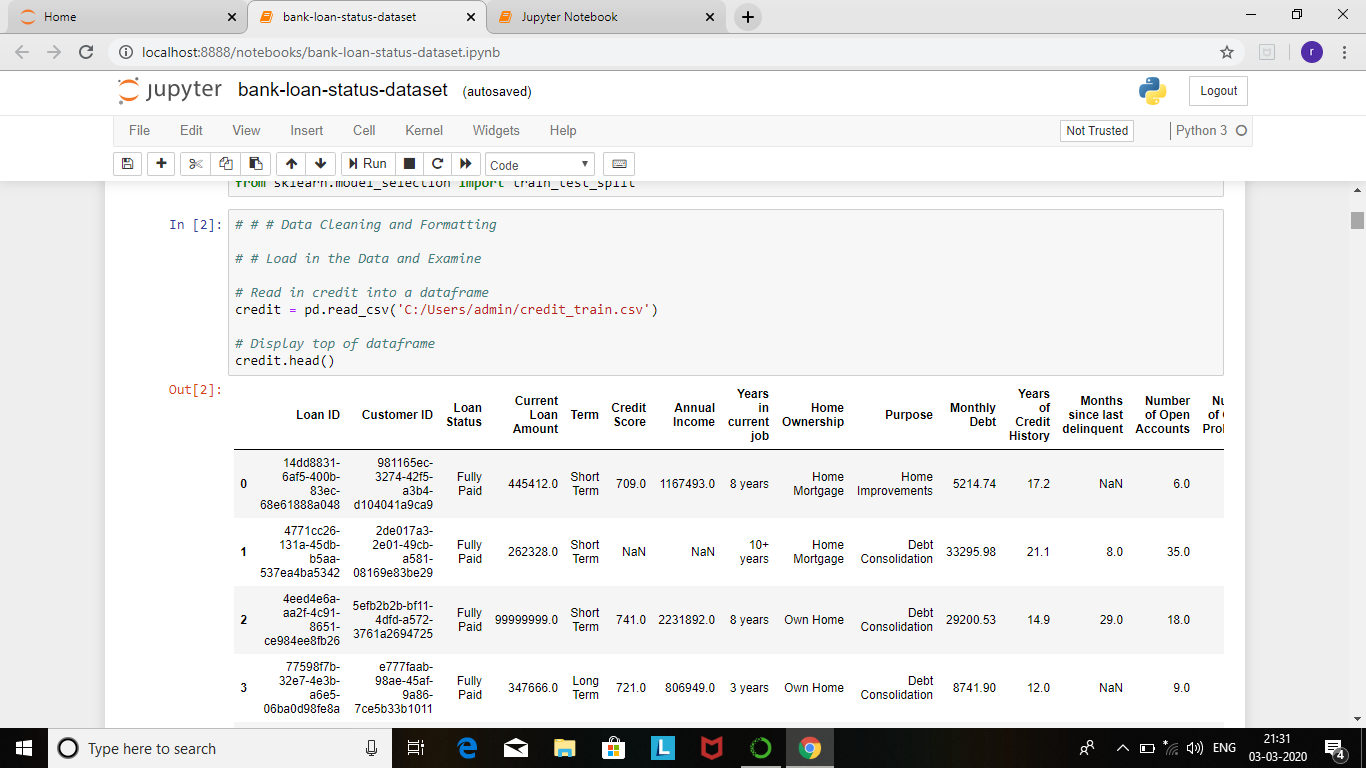
**# Splitting data into training and testing**

from sklearn.model\_selection import train\_test\_split



credit = pd.read\_csv('C:/Users/admin/credit\_train.csv')

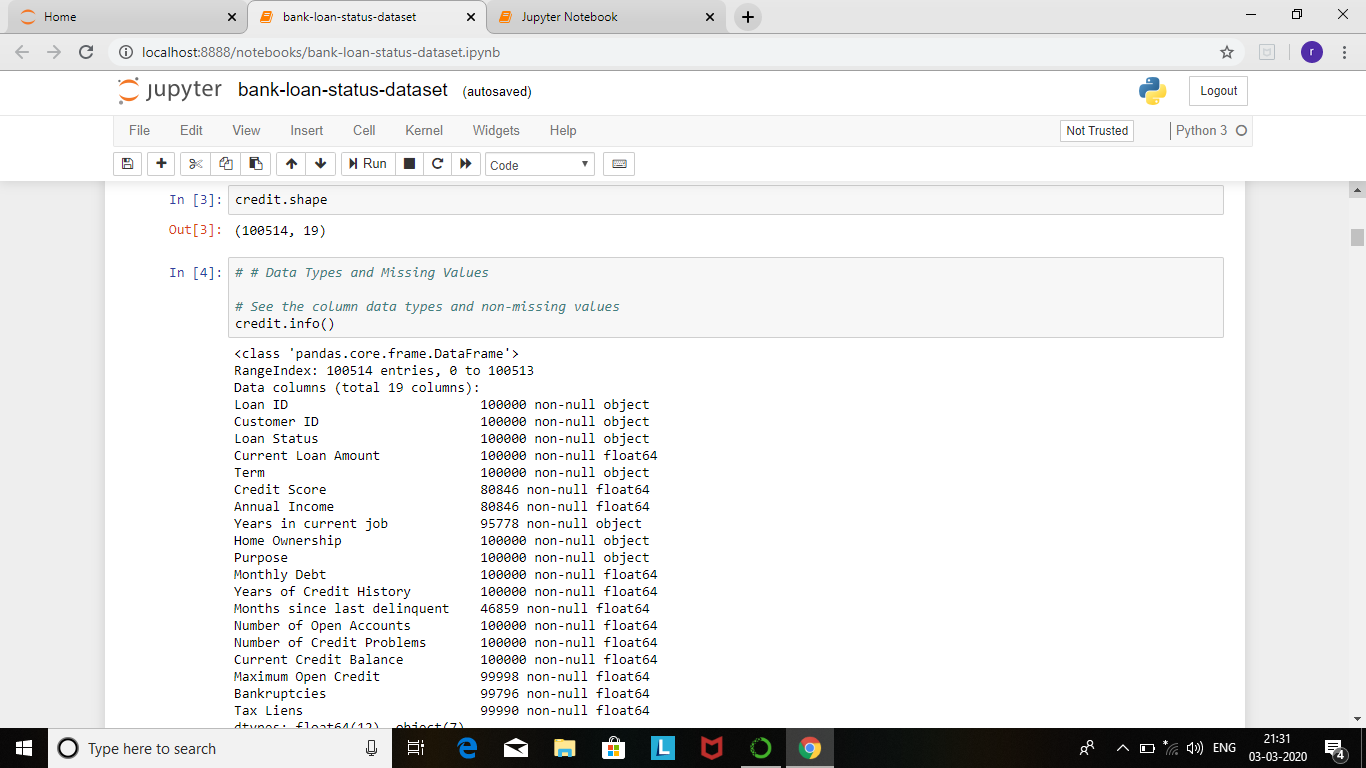
credit.head()

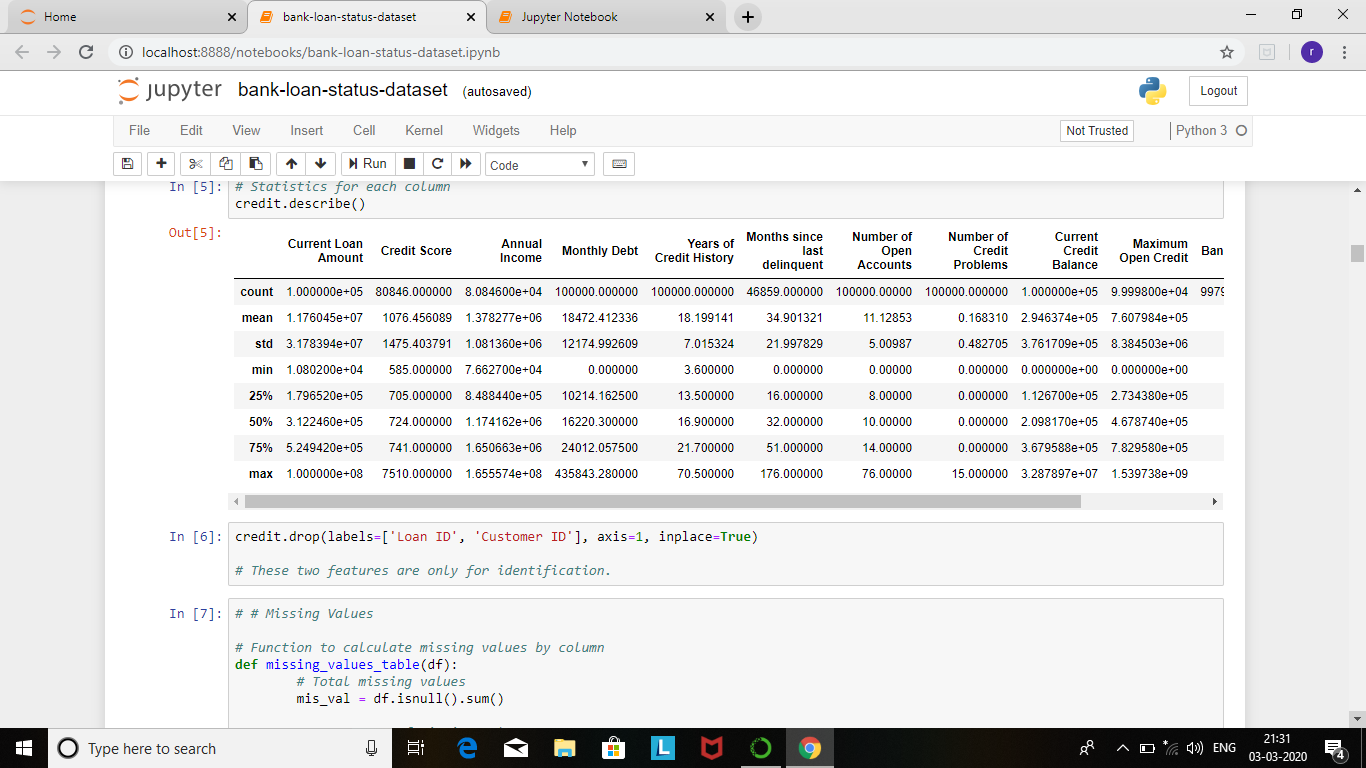


credit.shape

credit.info()

credit.describe()





credit.drop(labels=['Loan ID', 'Customer ID'], axis=1, inplace=True)

def missing\_values\_table(df):

**# Total missing values**

mis\_val = df.isnull().sum()

**# Percentage of missing values**

mis\_val\_percent = 100 \* df.isnull().sum() / len(df)

**# Make a table with the results**

mis\_val\_table = pd.concat([mis\_val, mis\_val\_percent], axis=1)

**# Rename the columns**

mis\_val\_table\_ren\_columns = mis\_val\_table.rename(

columns = {0 : 'Missing Values', 1 : '% of Total Values'})

**# Sort the table by percentage of missing descending**

mis\_val\_table\_ren\_columns = mis\_val\_table\_ren\_columns[

mis\_val\_table\_ren\_columns.iloc[:,1] != 0].sort\_values(

'% of Total Values', ascending=False).round(1)

**# Print some summary information**

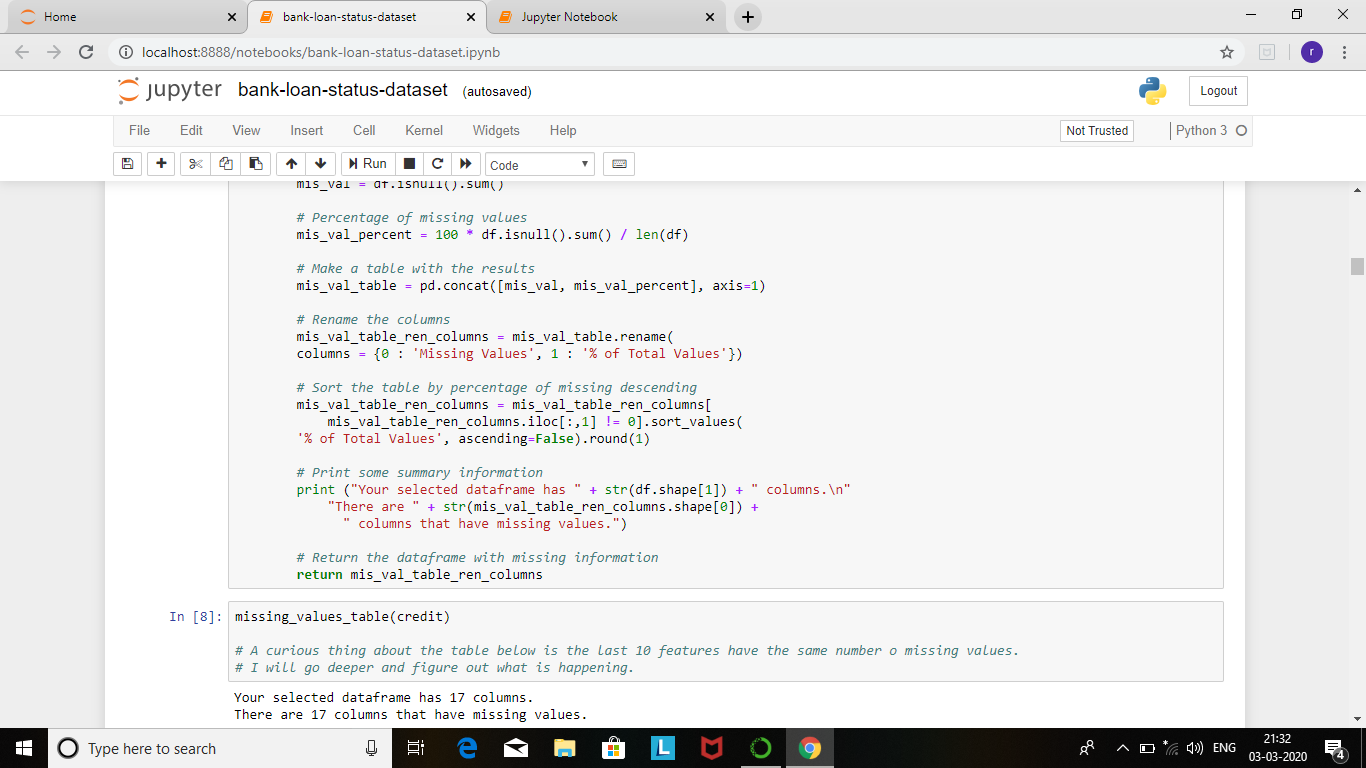
print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"

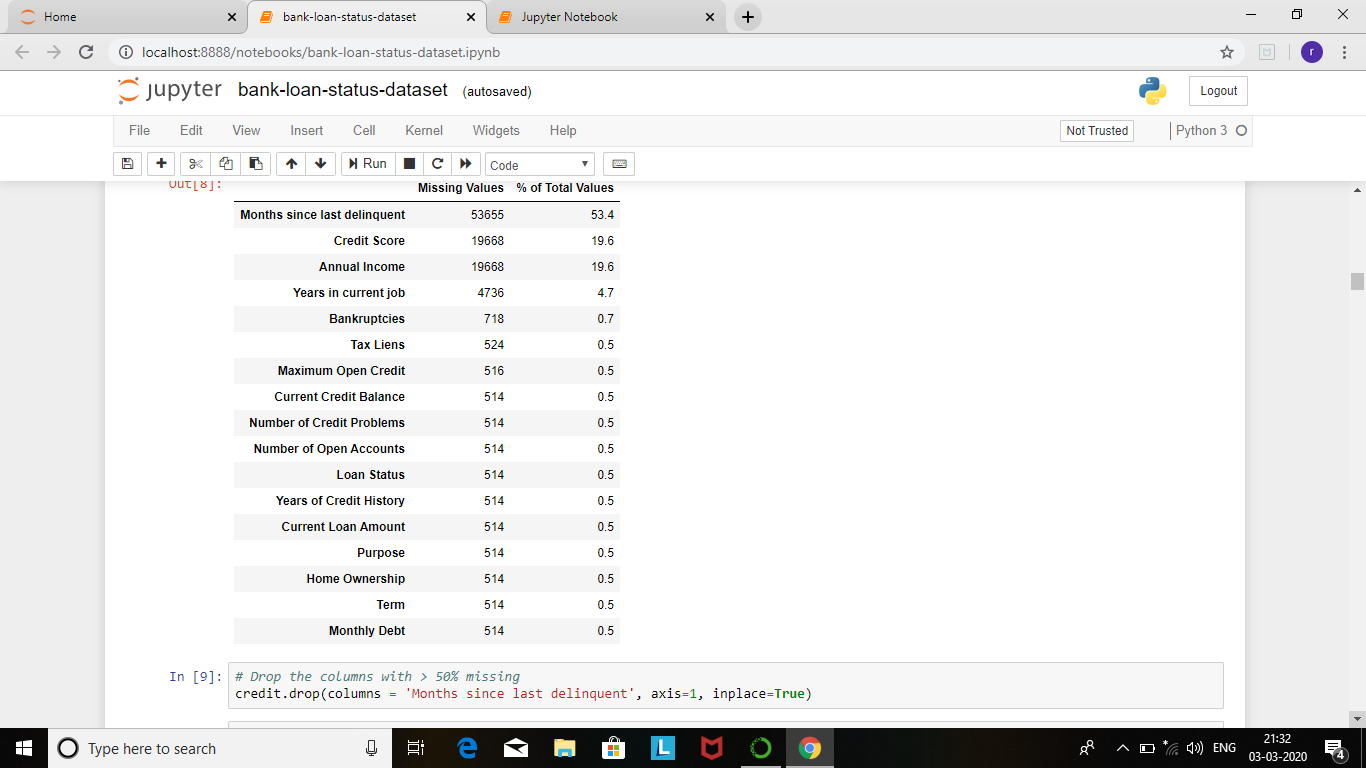
"There are " + str(mis\_val\_table\_ren\_columns.shape[0]) +

" columns that have missing values.")

**# Return the dataframe with missing information**

return mis\_val\_table\_ren\_columns



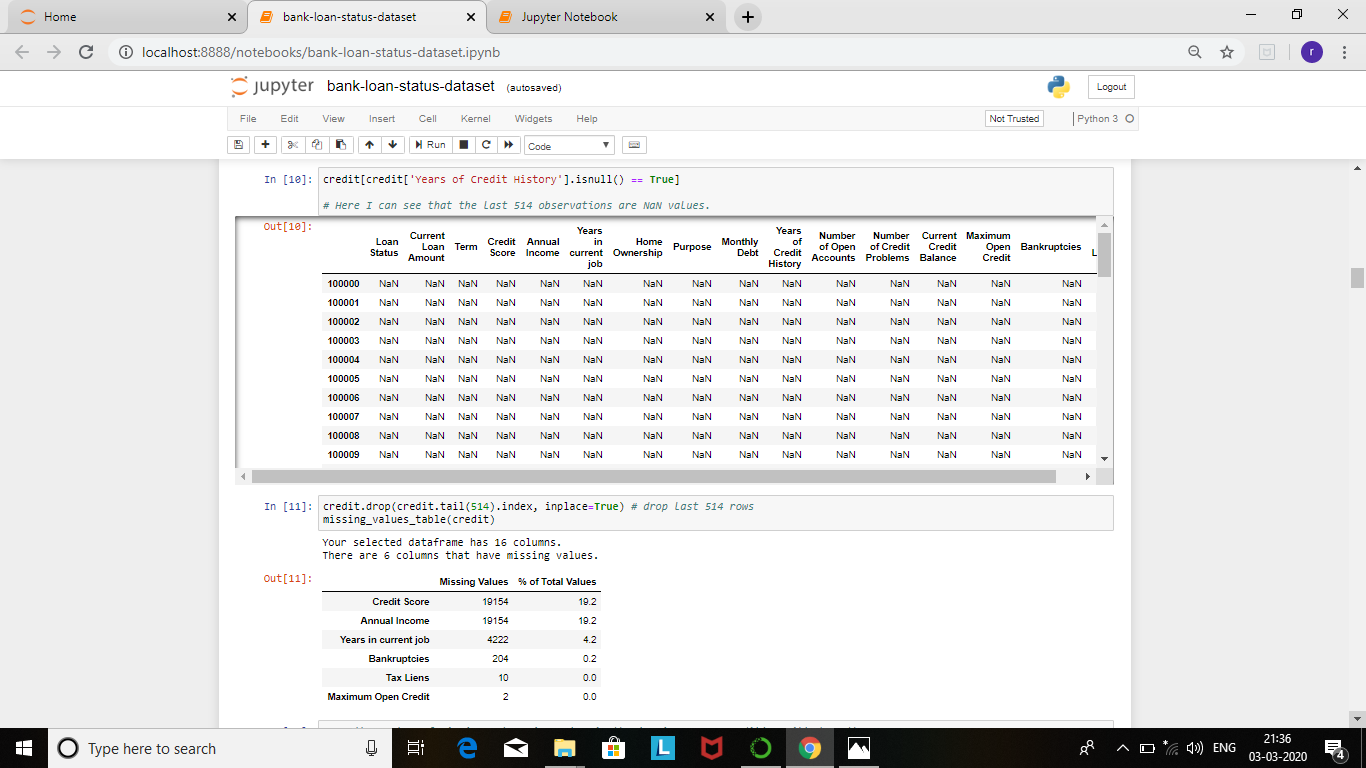


credit.drop(columns = 'Months since last delinquent', axis=1, inplace=True)

credit[credit['Years of Credit History'].isnull() == True]

credit.drop(credit.tail(514).index, inplace=True) **# drop last 514 rows**

missing\_values\_table(credit)



**# As the number of missing values is so low in the 'Maximum Open Credit' I will drop them**.

for i in credit['Maximum Open Credit'][credit['Maximum Open Credit'].isnull() == True].index:

credit.drop(labels=i, inplace=True)

missing\_values\_table(credit)

**# As the number of missing values is so low in the 'Tax Liens' I will drop them.**

for i in credit['Tax Liens'][credit['Tax Liens'].isnull() == True].index:

credit.drop(labels=i, inplace=True)

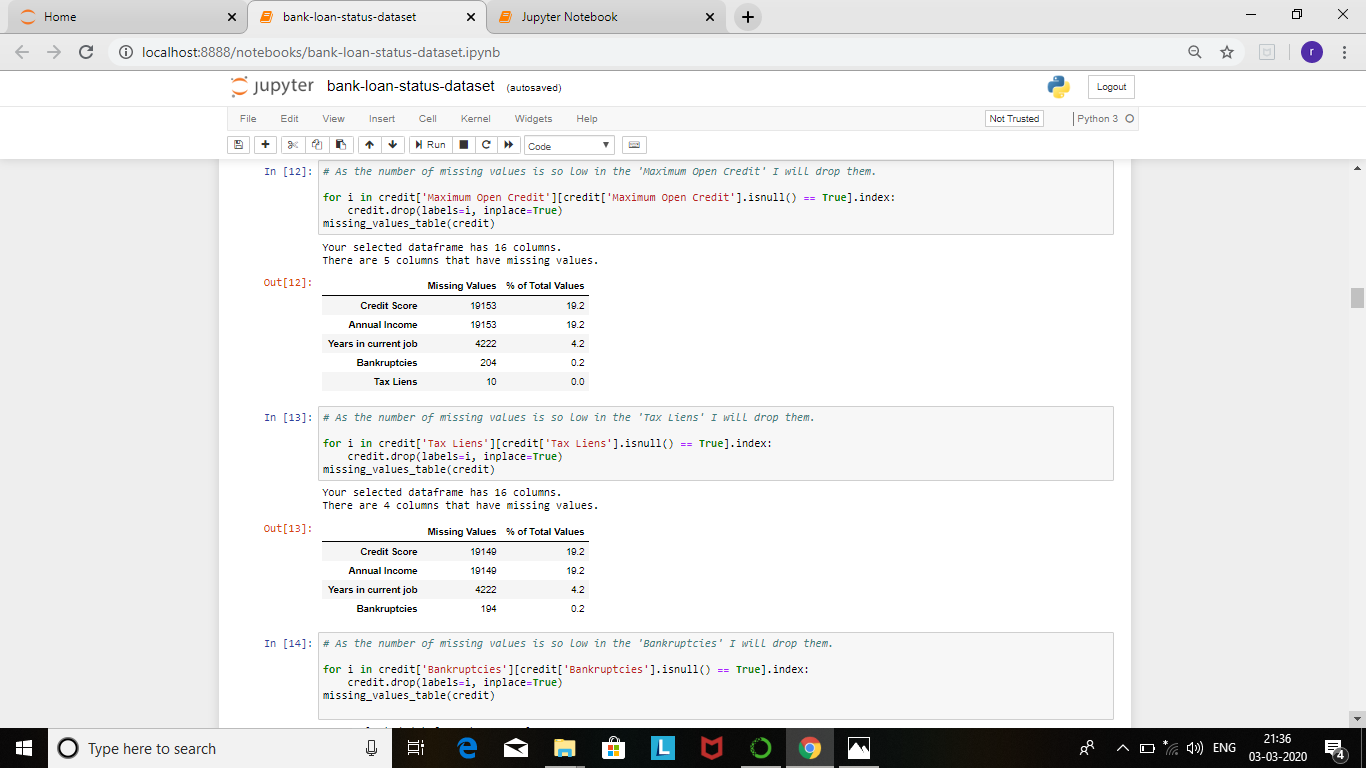
missing\_values\_table(credit)

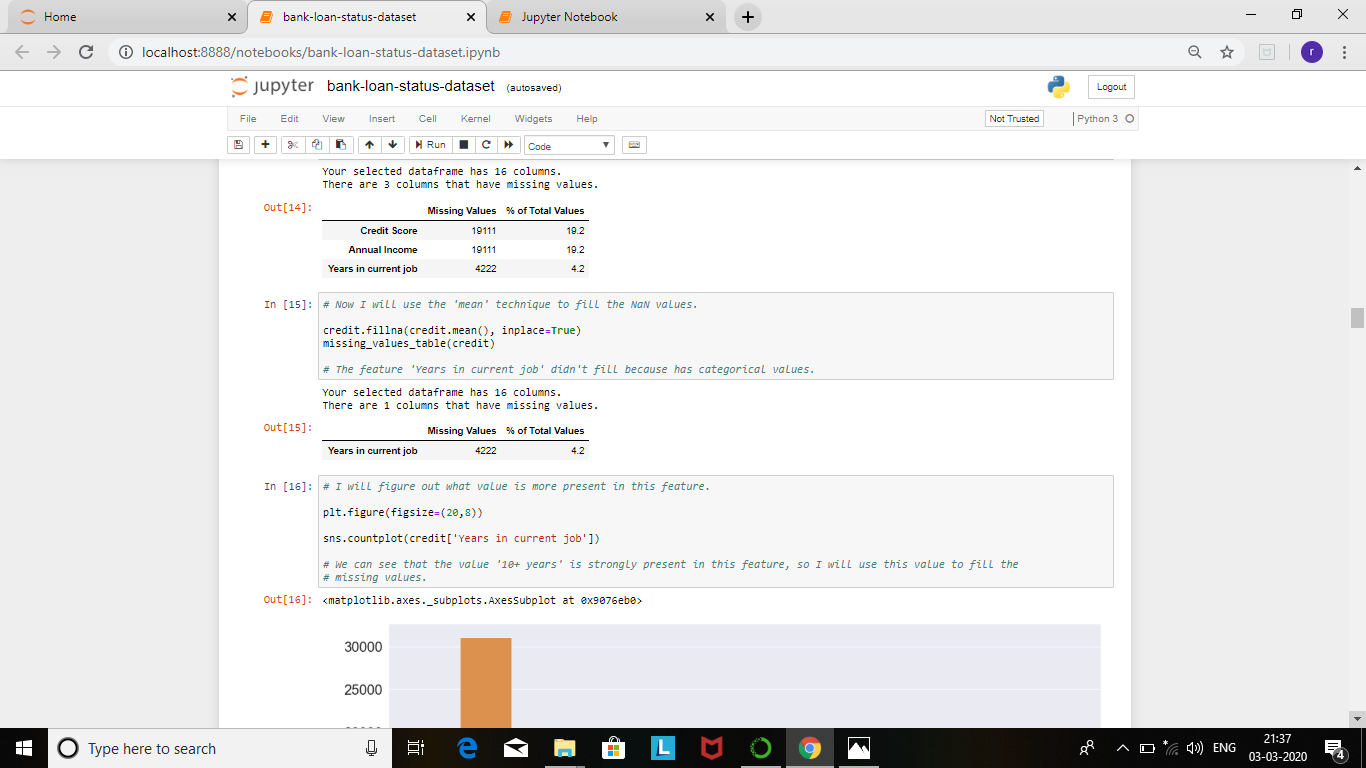
# As the number of missing values is so low in the 'Bankruptcies' I will drop them.

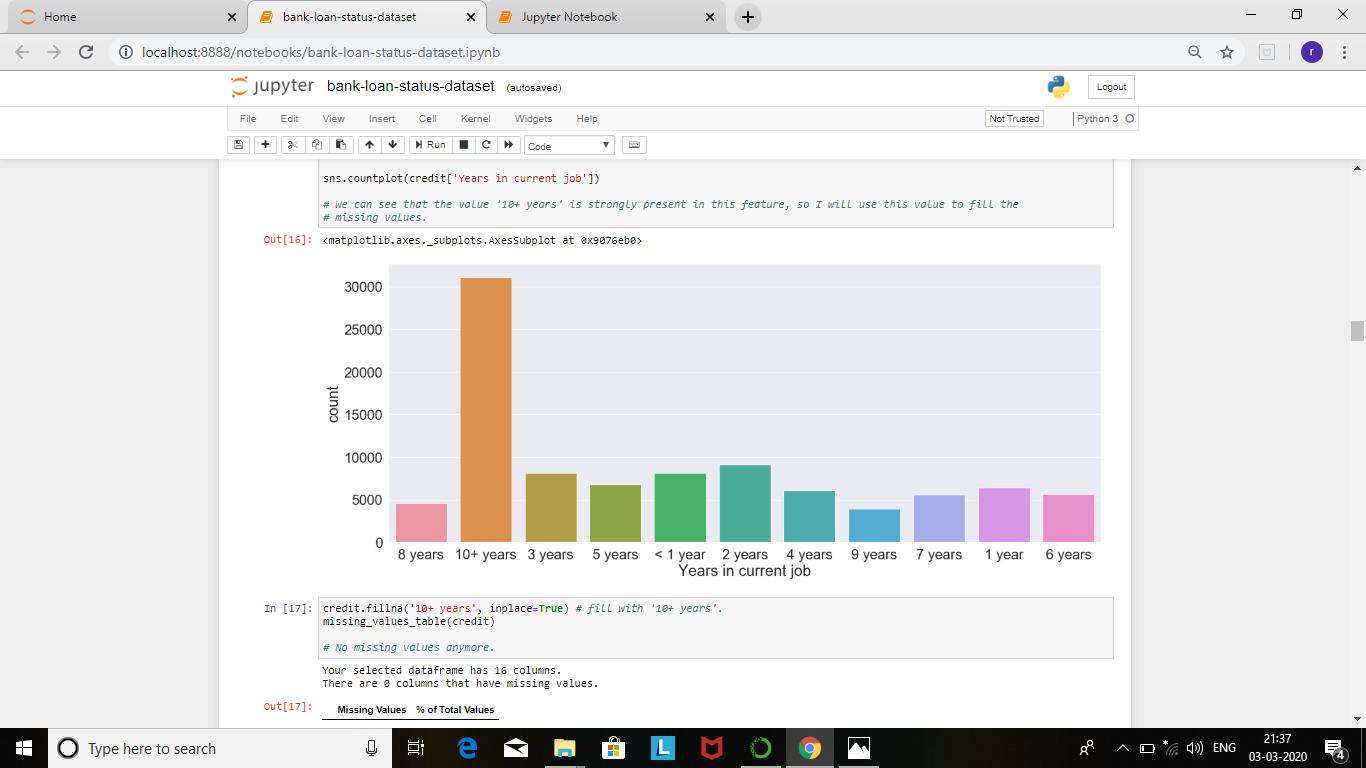
for i in credit['Bankruptcies'][credit['Bankruptcies'].isnull() == True].index:

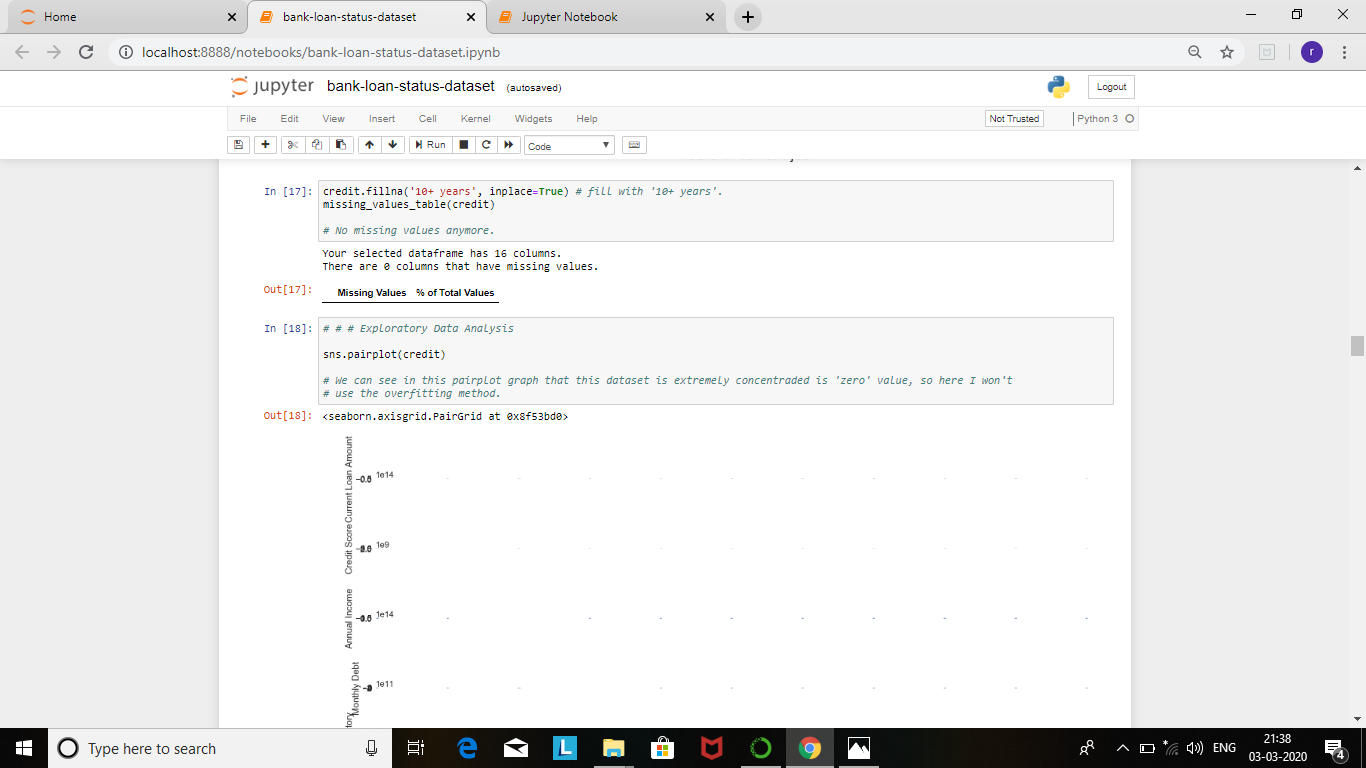
credit.drop(labels=i, inplace=True)

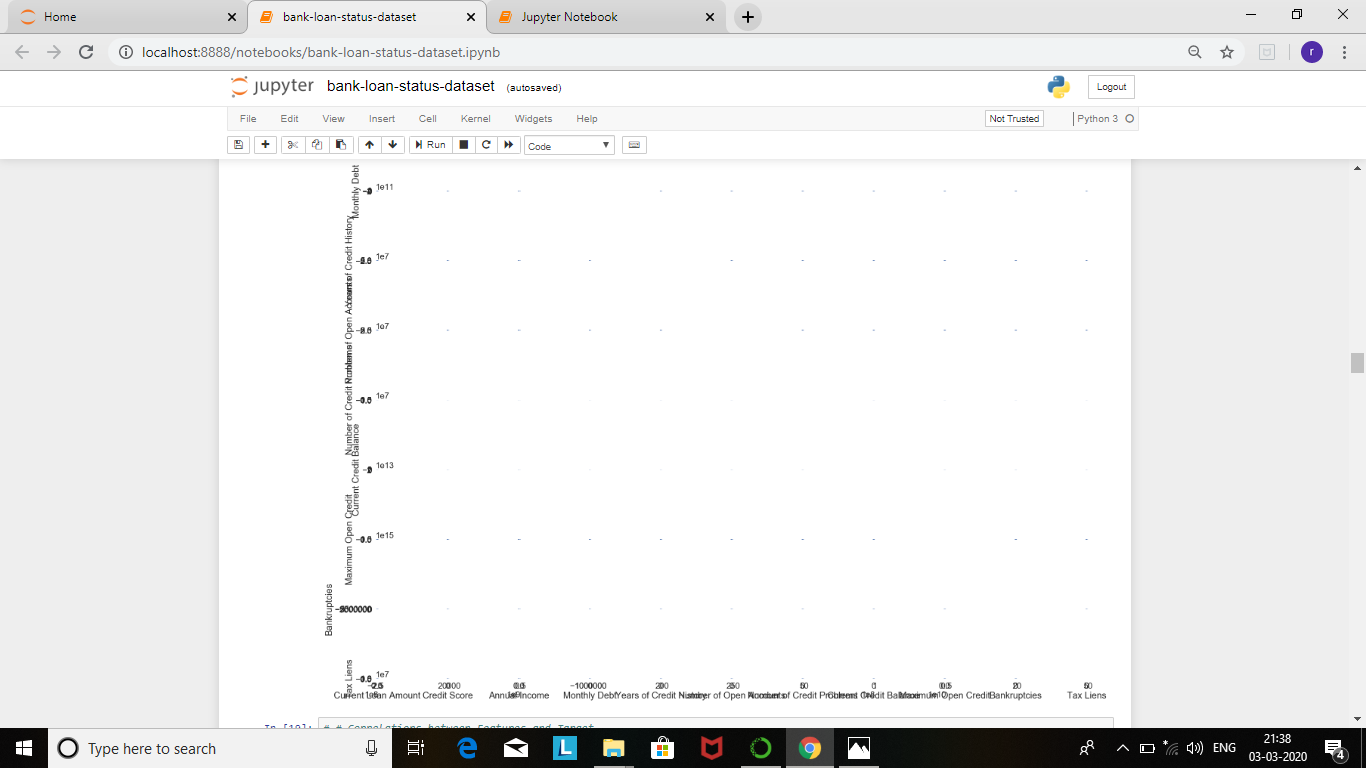
missing\_values\_table(credit)

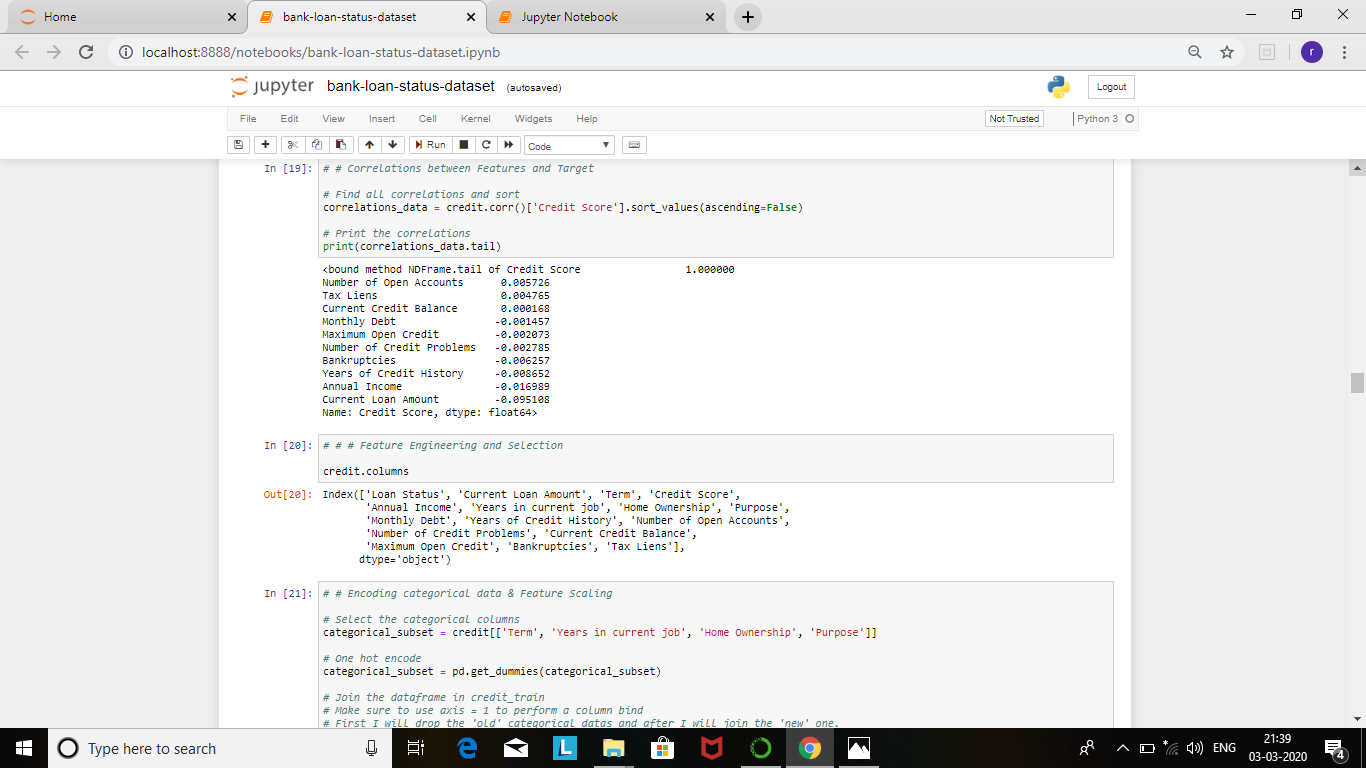


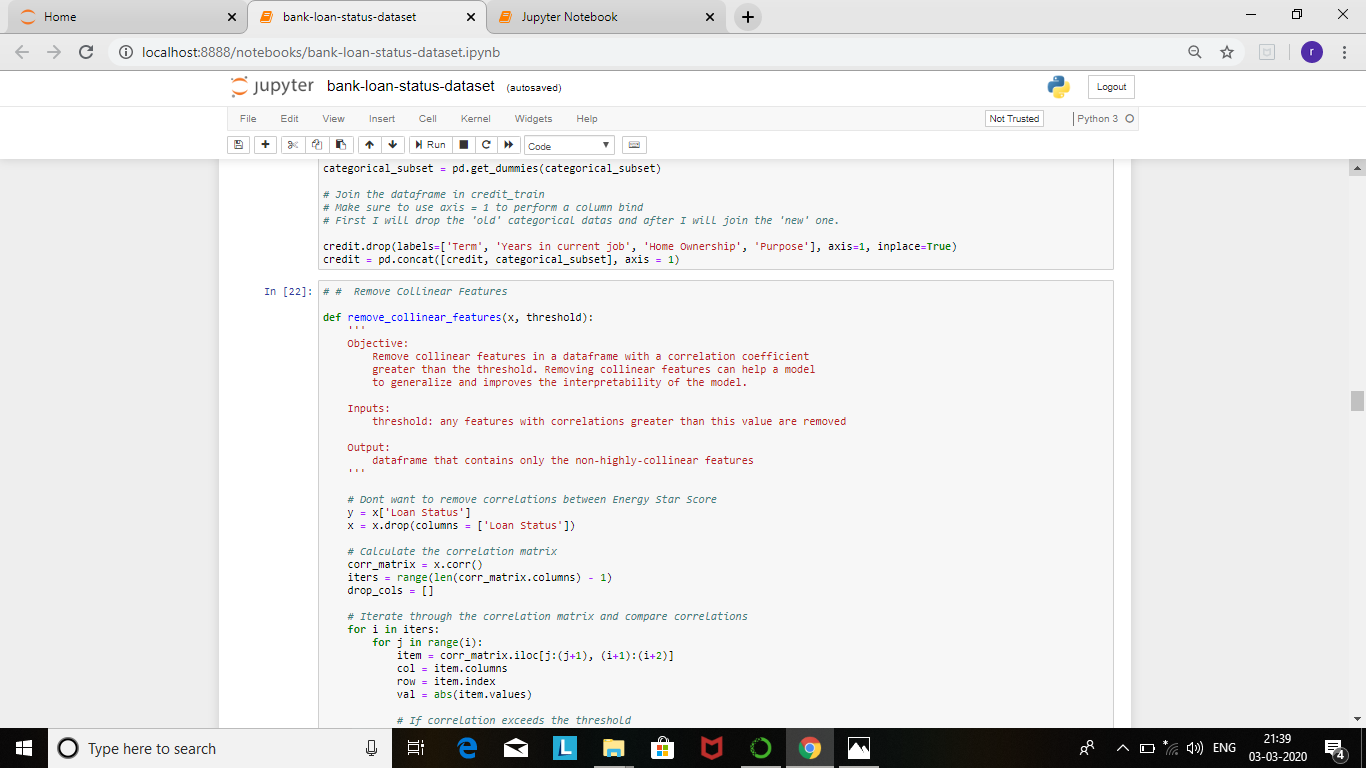


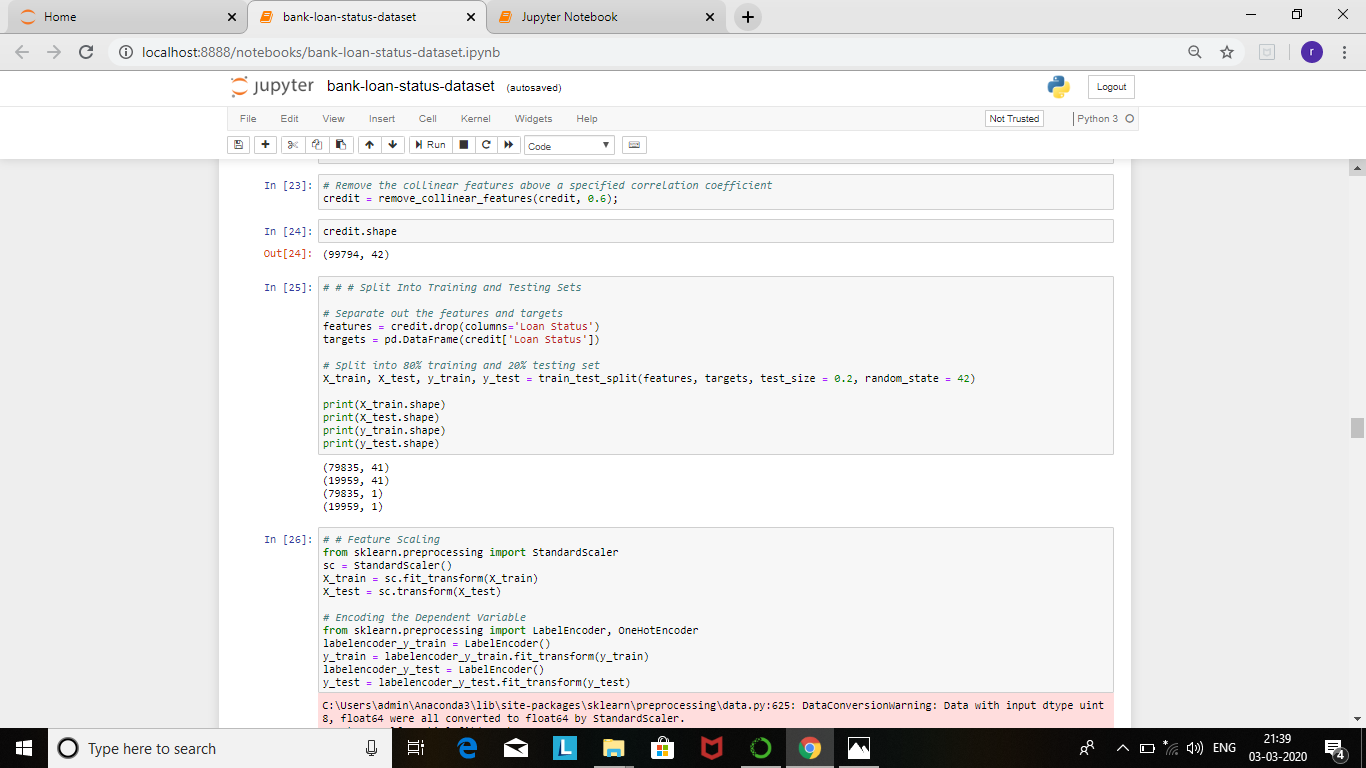


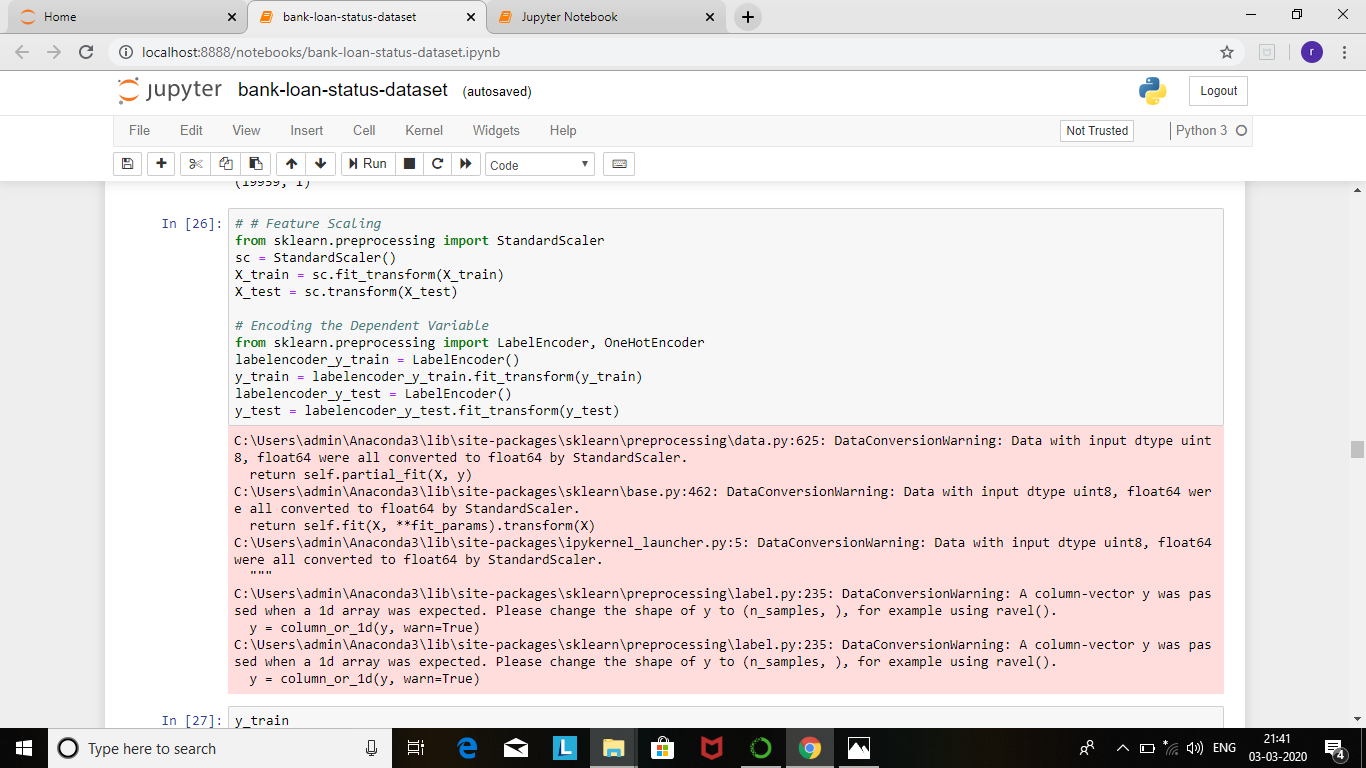


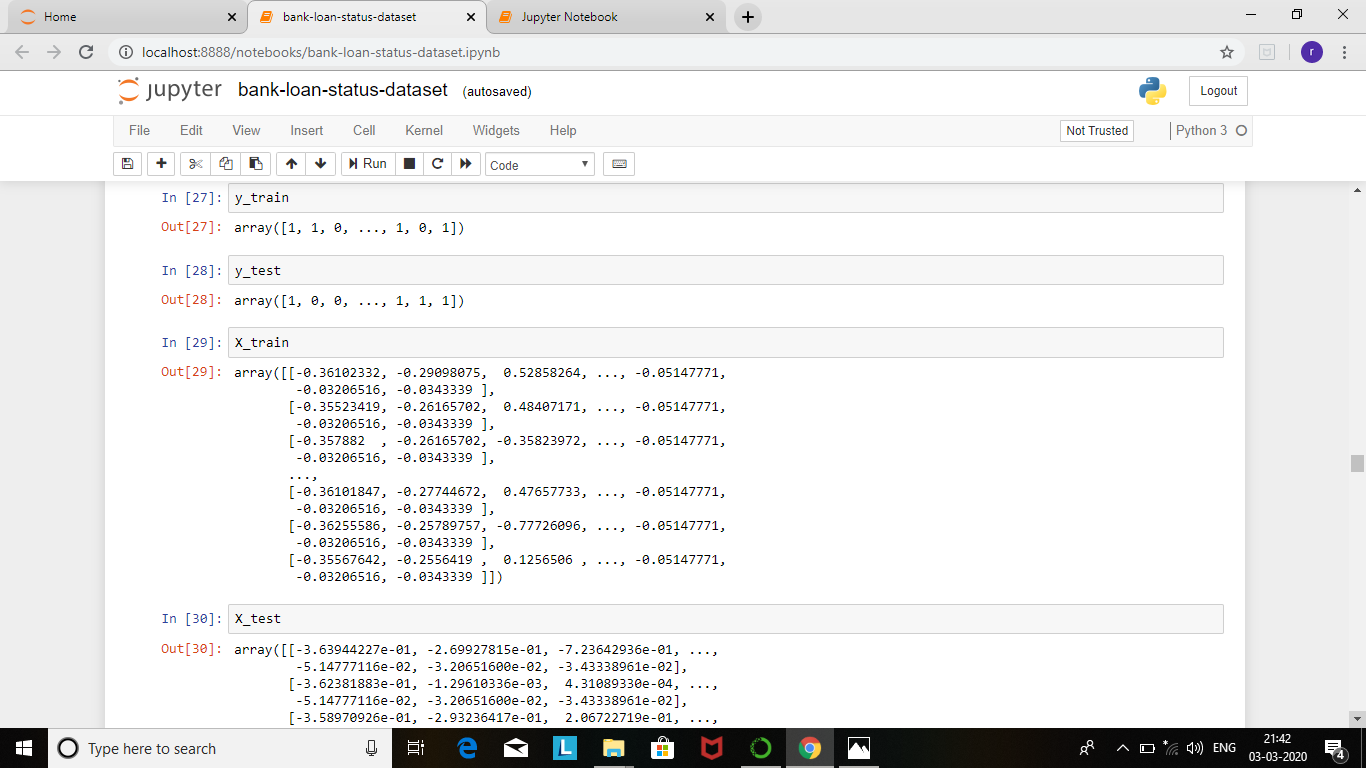


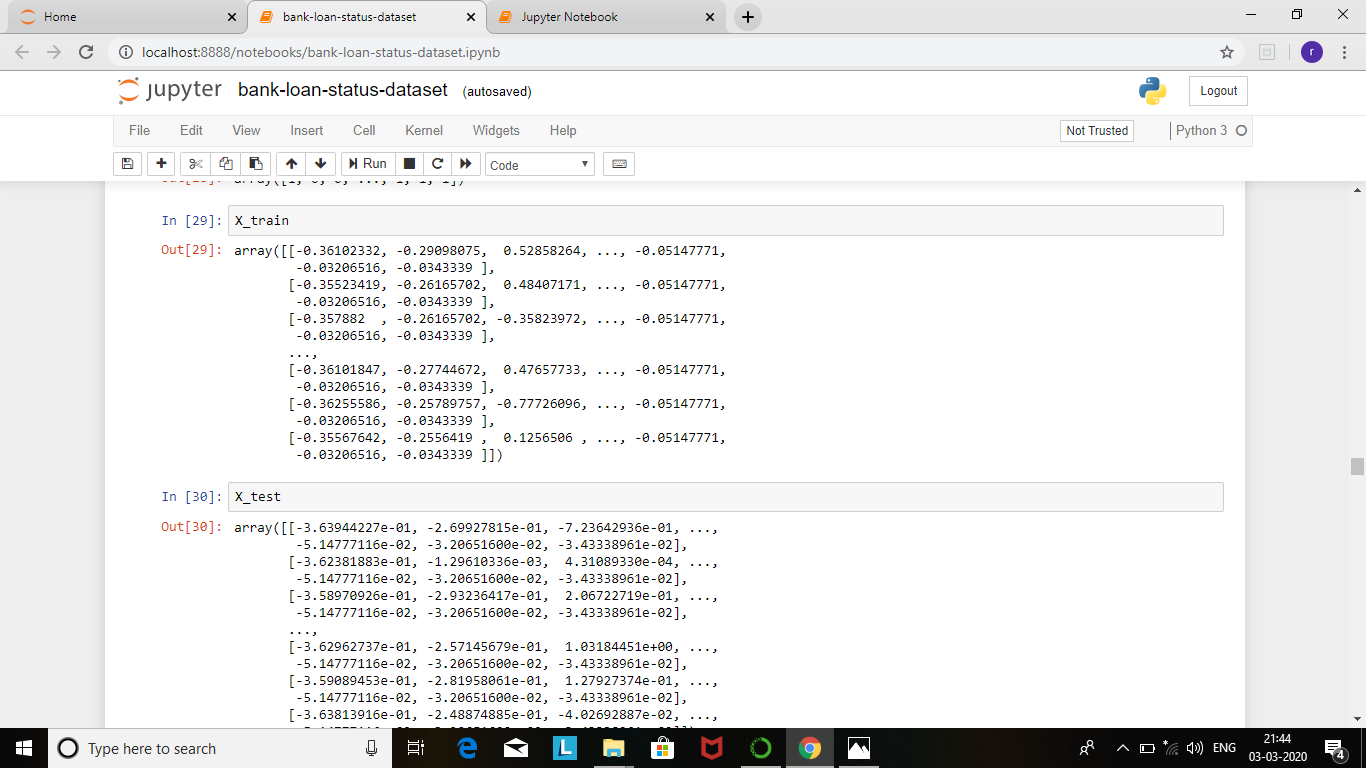


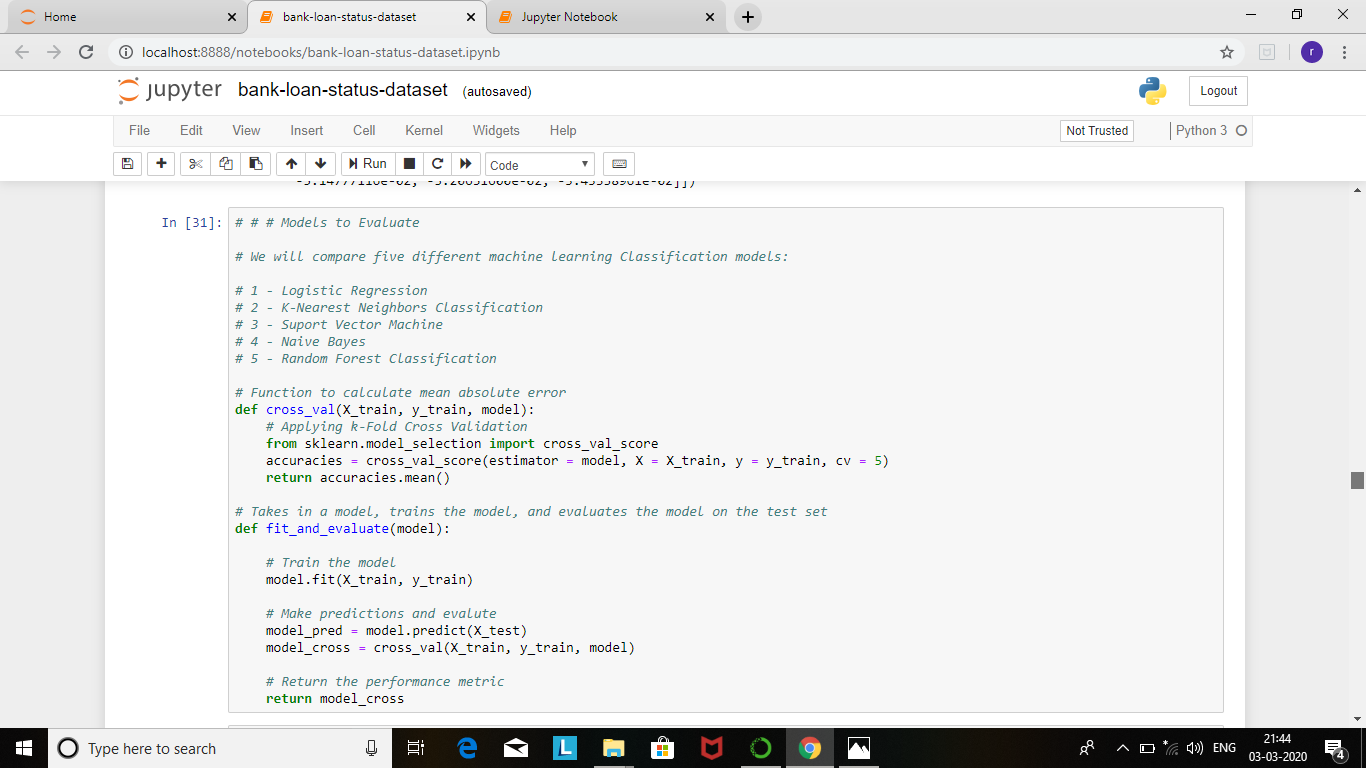


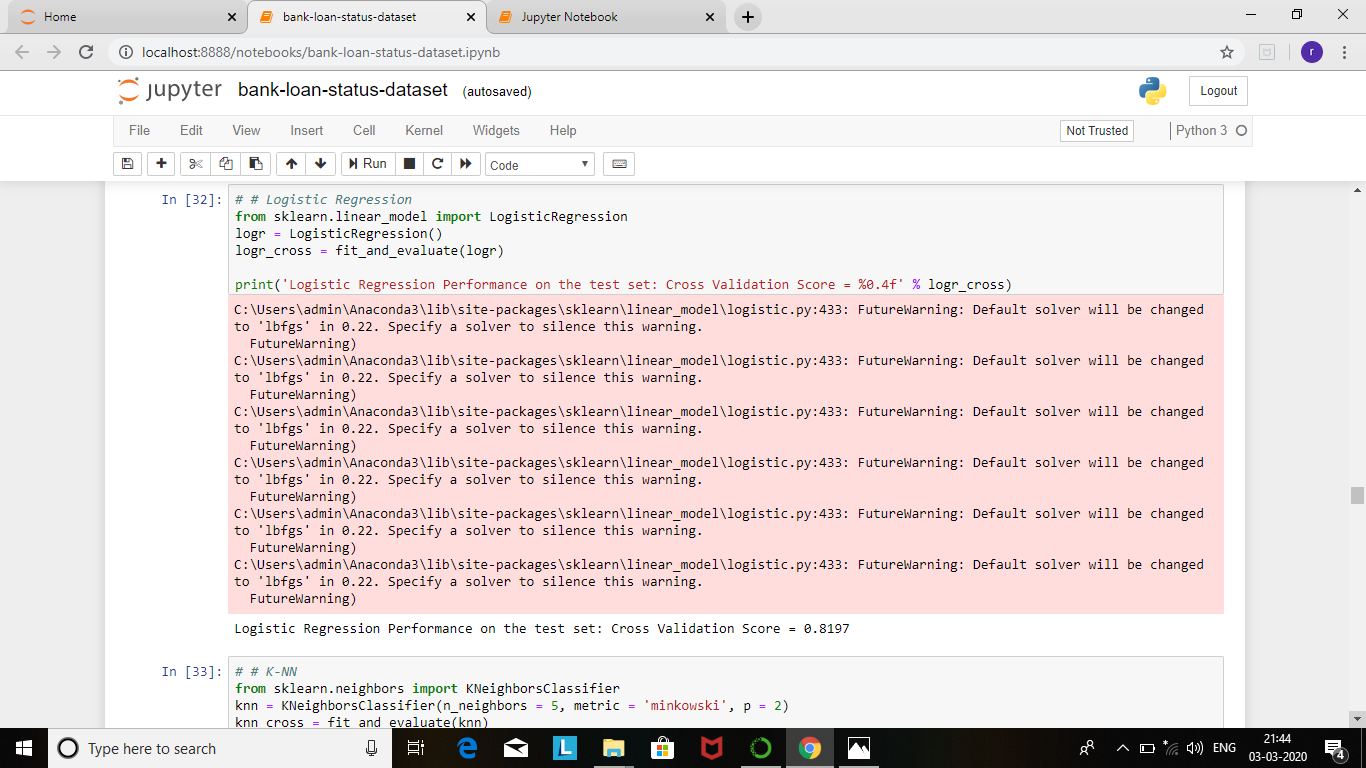


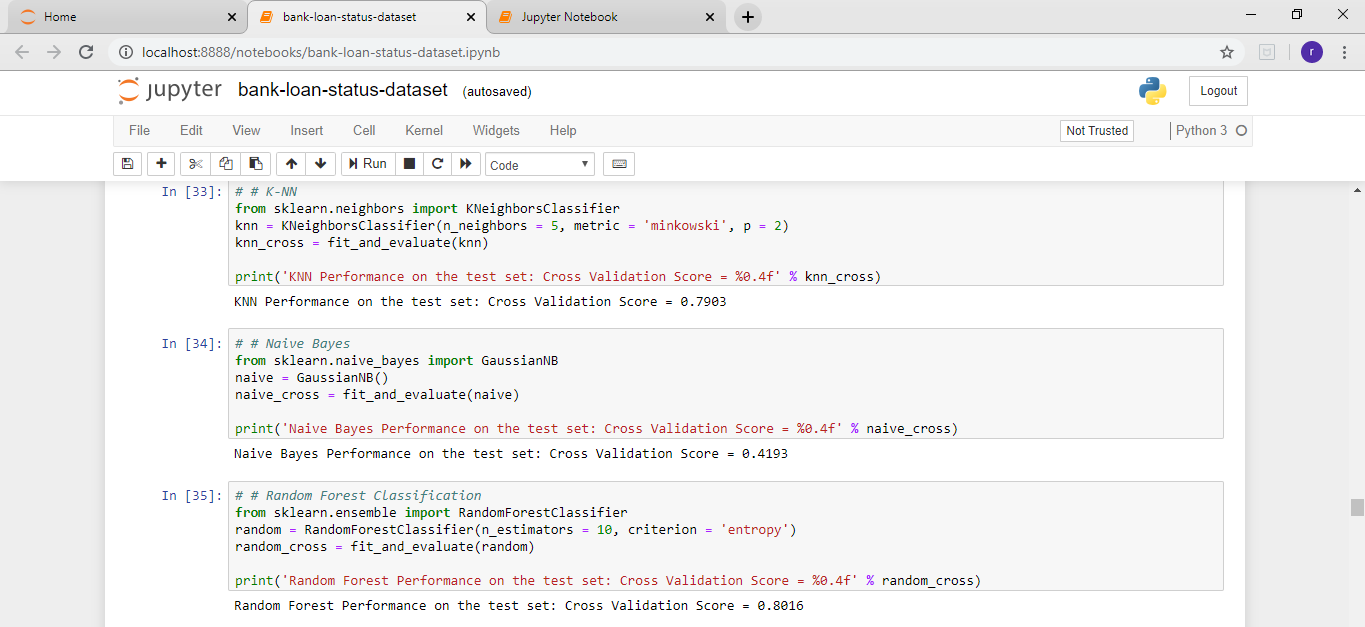












**CONCLUSION**

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. This application is working properly and meeting to all Banker requirements. This component can be easily plugged in many other systems. There have been numbers cases of computer glitches, errors in content and most important weight of features is fixed in automated prediction system, So in the near future the so –called software could be made more secure, reliable and dynamic weight adjustment .In near future this module of prediction can be integrate with the module of automated processing system. the system is trained on old training dataset in future software can be made such that new testing date should also take part in training data after some fix time.

URL: